BSDS ASSIGNMENT-2

Github Repo :

https://github.com/manika0407/bsdscourse/tree/master/Assignment2/bsdsassignment

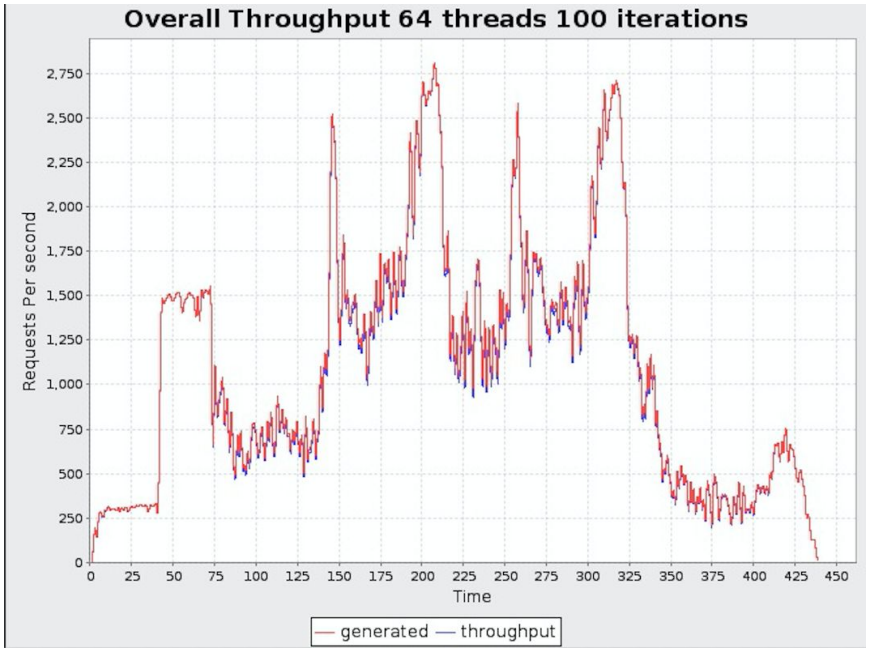
Step 1 and 2: Plot and performance statistics showing results for a test run with default clients (64) and settings for threads test.

Setup for single ec2 testing ->

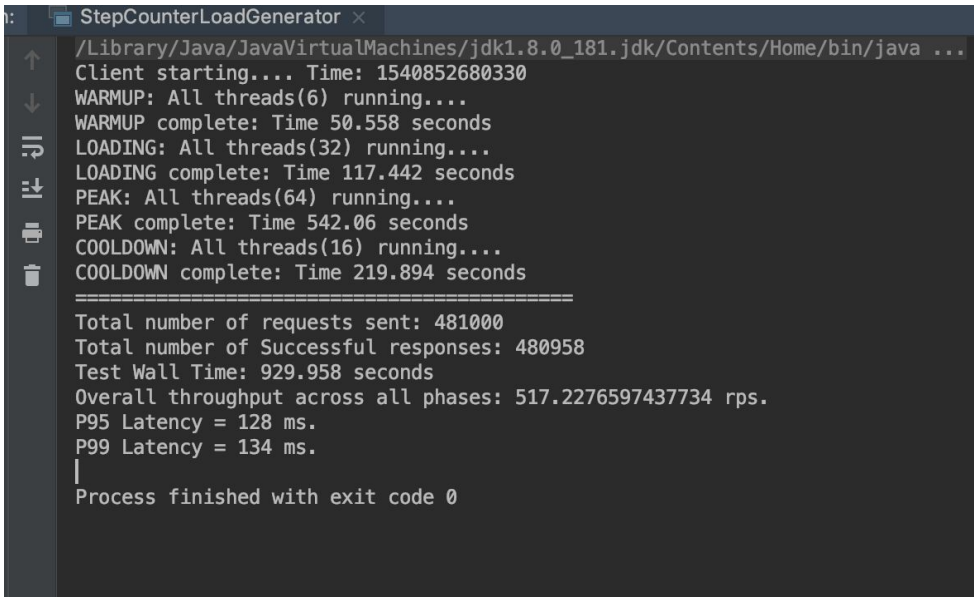
change tomcat server threads to use 256 thread executor
change default connection pool size to rds = 256
increase default Xmx settings for tomcat to 512M (although not required this high, but at default on 256 threads tomcat was running out of default java heap space)

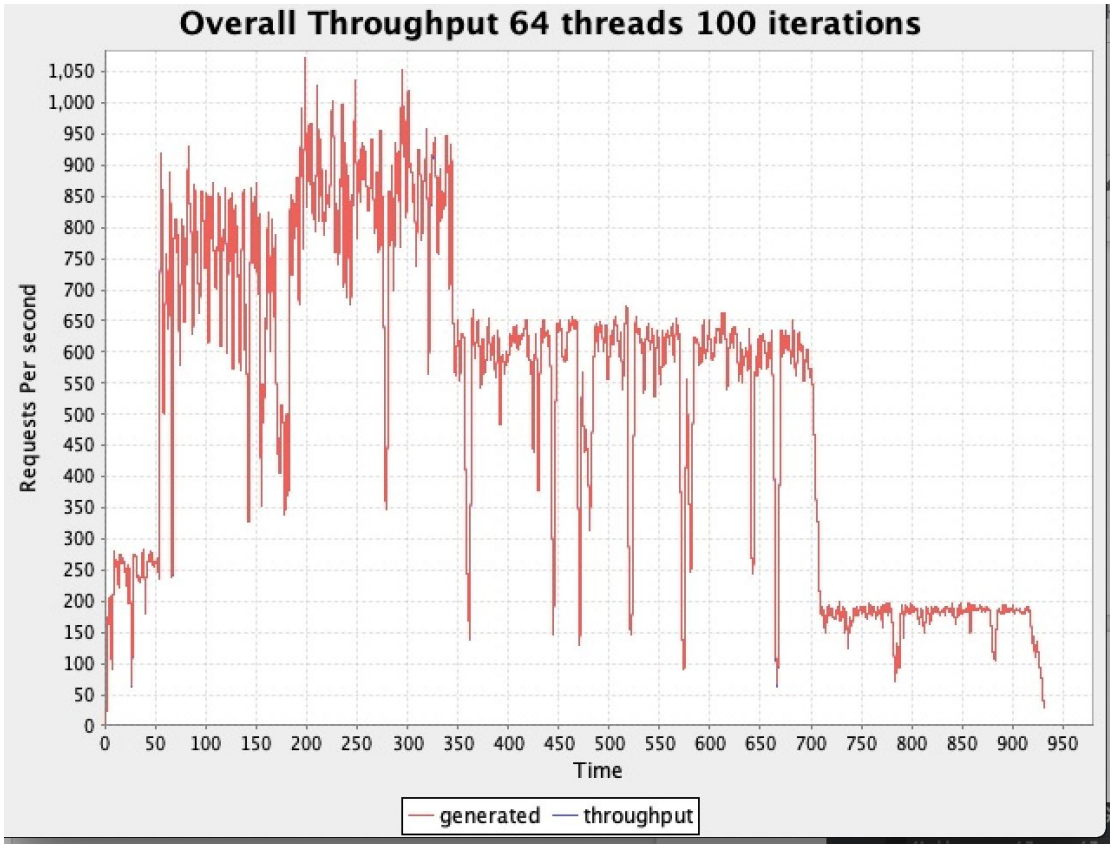
Cmd ->
java -jar bsdsassignment2-stress-jar-with-dependencies.jar -h "ec2-34-220-61-97.us-west-2.compute.amazonaws.com" -t 64 -g results/StepCounterThroughput_64_ec2.jpeg

Client starting.... Time: 1540924419155
WARMUP: All threads(6) running....
WARMUP complete: Time 39.483 seconds
LOADING: All threads(32) running....
LOADING complete: Time 92.21 seconds
PEAK: All threads(64) running....
PEAK complete: Time 228.166 seconds
COOLDOWN: All threads(16) running....
COOLDOWN complete: Time 77.127 seconds
==========
Total number of requests sent: 481000
Total number of Successful responses: 473718
Test Wall Time: 436.988 seconds
Overall throughput across all phases: 1100.7167244867137 rps.
P95 Latency = 26 ms.
P99 Latency = 28 ms.



Note:- The plot and statistics that I pasted above were the results i got the first time. Next time I ran the tests for same 64 threads, I got totally different results (maybe due to some network performance variance, tested this on multiple client machines or ec2 micro tier machines has bandwidth limitation per window.)

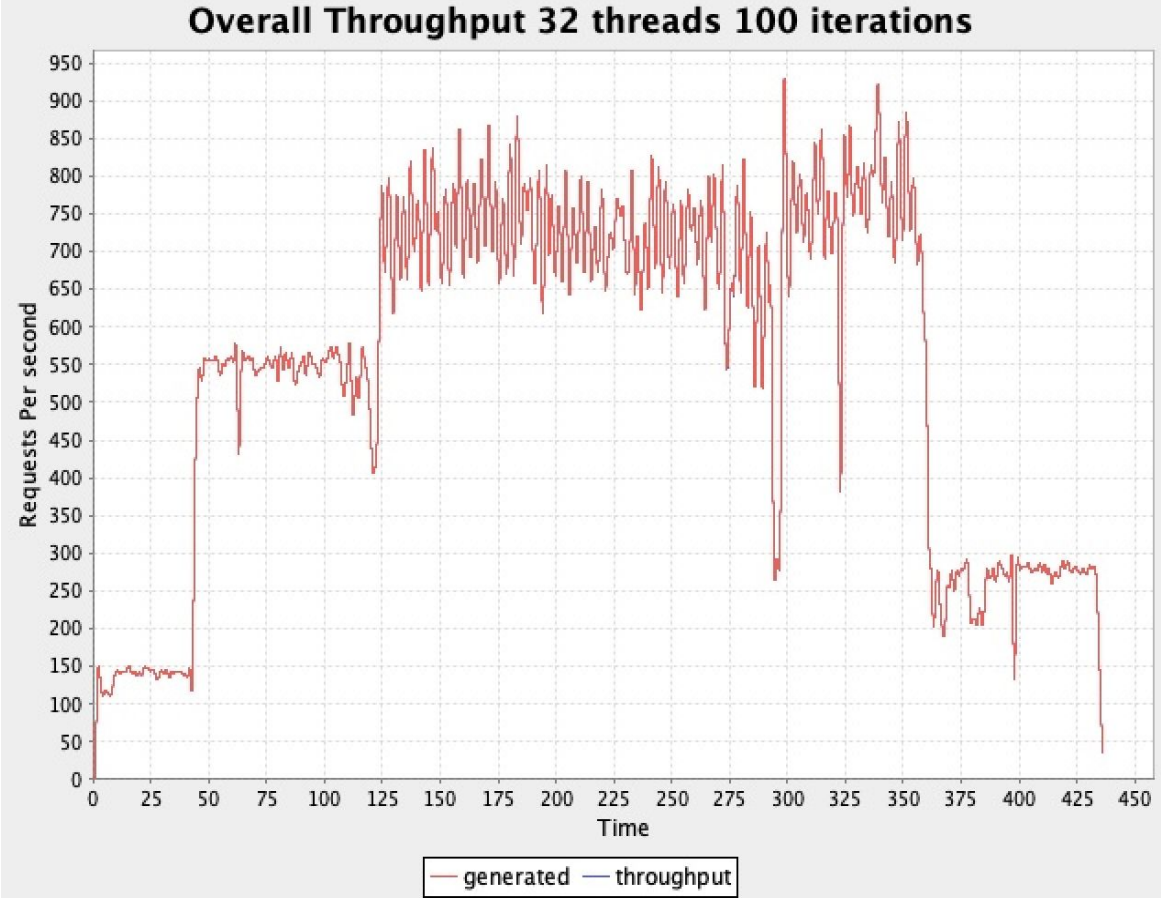



Overall Throughput 64 threads 100 iterations
Requests Per second
1,050
1,000
950
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
0 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950
Time
generated
throughput

Step 3 : Plots and statistics for the 4 tests run against a single server

Case 1 : 32 threads against a single server

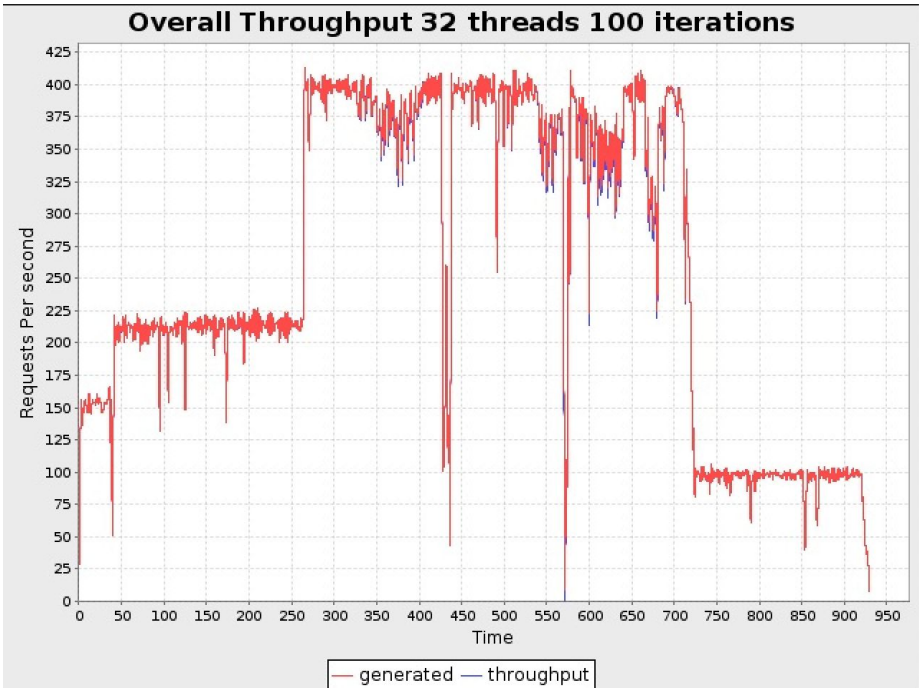
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ..
Client starting.... Time: 1540849954653
WARMUP: All threads(3) running....
WARMUP complete: Time 41.411 seconds
LOADING: All threads(16) running....
LOADING complete: Time 76.267 seconds
PEAK: All threads(32) running....
PEAK complete: Time 242.686 seconds
COOLDOWN: All threads(8) running....
COOLDOWN complete: Time 74.158 seconds
================================================
Total number of requests sent: 240500
Total number of Successful responses: 240496
Test Wall Time: 434.526 seconds
Overall throughput across all phases: 553.4766619258686 rps.
P95 Latency = 35 ms.
P99 Latency = 43 ms.

Process finished with exit code 0
```



Note:- The plot and statistics that I pasted above were the results i got the first time. Next time I ran the tests for 32 threads, I got totally different results (maybe due to some network performance variance, tested this on multiple client machines)

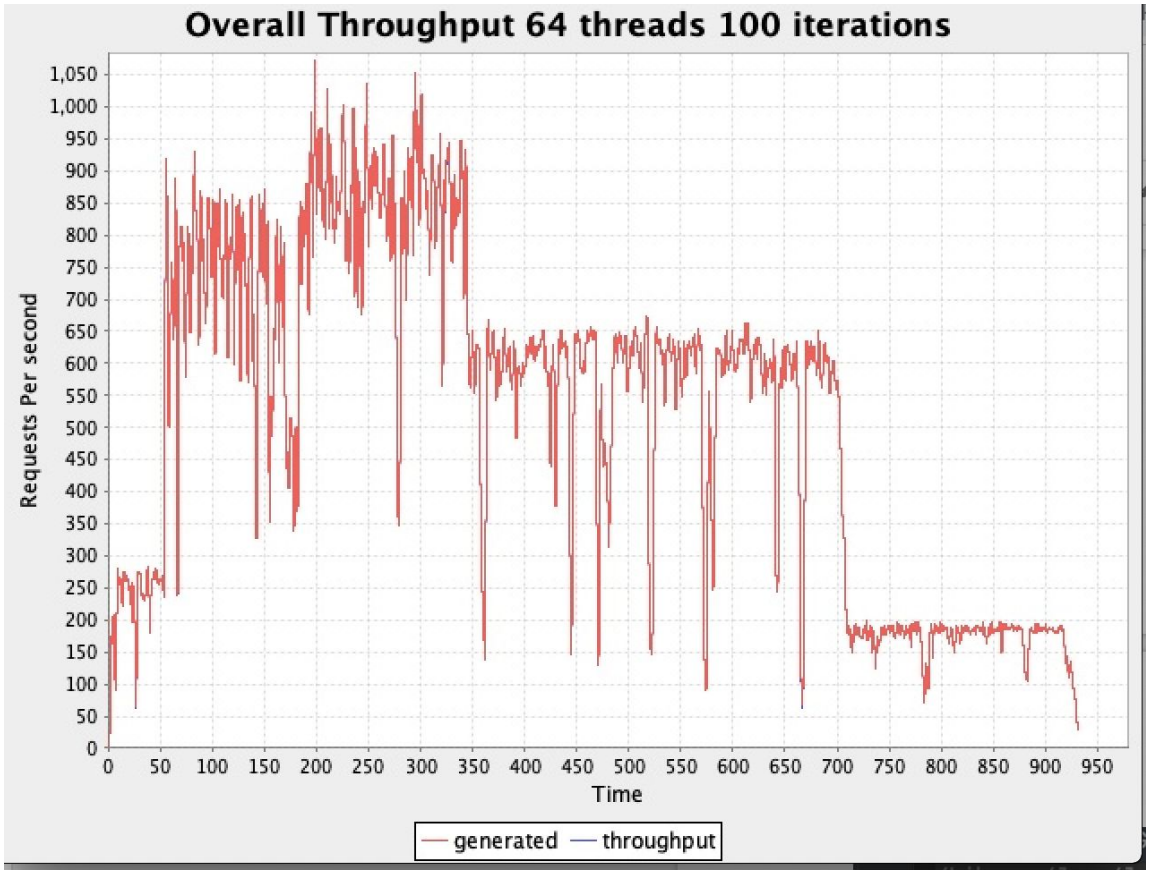
32 threads

Client starting.... Time: 1540888638914
WARMUP: All threads(3) running....
WARMUP complete: Time 36.944 seconds
LOADING: All threads(16) running....
LOADING complete: Time 205.46 seconds
PEAK: All threads(32) running....
PEAK complete: Time 480.433 seconds
COOLDOWN: All threads(8) running....
COOLDOWN complete: Time 204.83 seconds
==========
Total number of requests sent: 240500
Total number of Successful responses: 239896
Test Wall Time: 927.67 seconds
Overall throughput across all phases: 259.25167354770554 rps.
P95 Latency = 81 ms.
P99 Latency = 85 ms.



Case 2: 64 threads against a single server

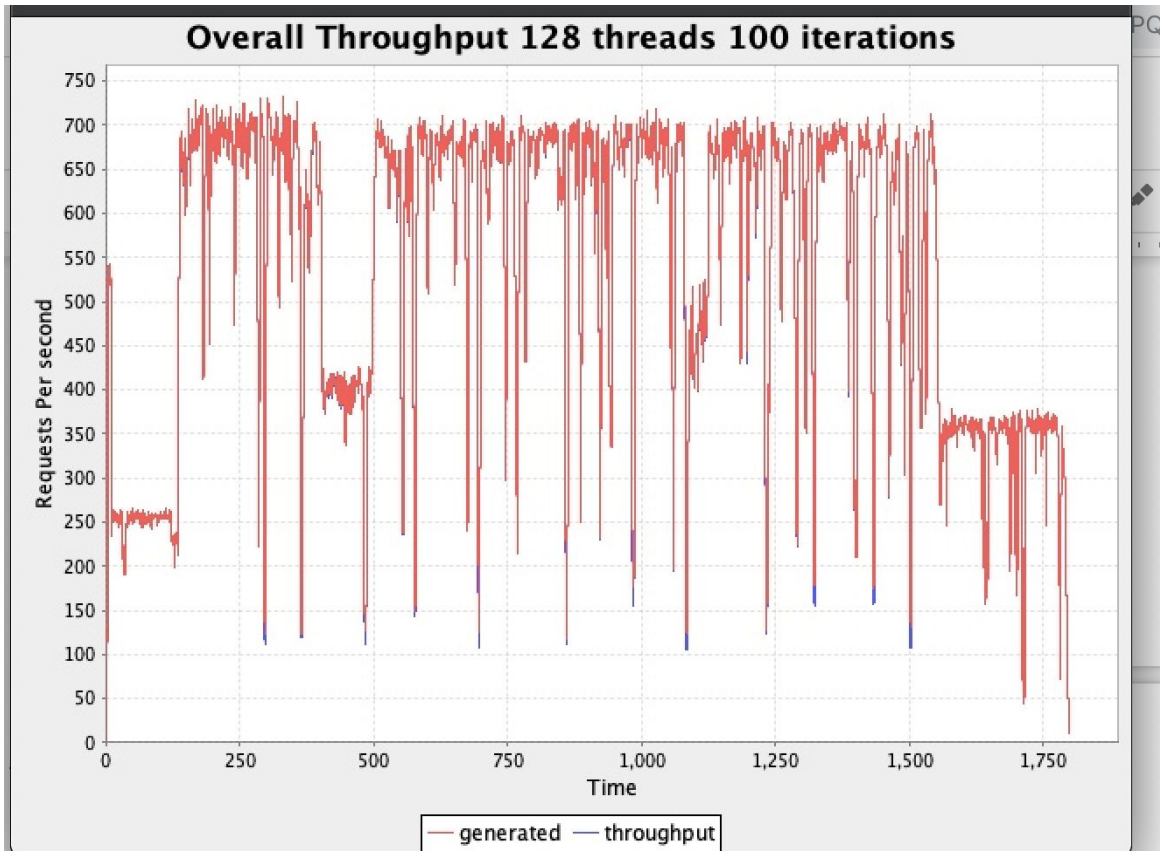
```
StepCounterLoadGenerator
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...
Client starting.... Time: 1540852680330
WARMUP: All threads(6) running....
WARMUP complete: Time 50.558 seconds
LOADING: All threads(32) running....
LOADING complete: Time 117.442 seconds
PEAK: All threads(64) running....
PEAK complete: Time 542.06 seconds
COOLDOWN: All threads(16) running....
COOLDOWN complete: Time 219.894 seconds
================================================
Total number of requests sent: 481000
Total number of Successful responses: 480958
Test Wall Time: 929.958 seconds
Overall throughput across all phases: 517.2276597437734 rps.
P95 Latency = 128 ms.
P99 Latency = 134 ms.

Process finished with exit code 0
```



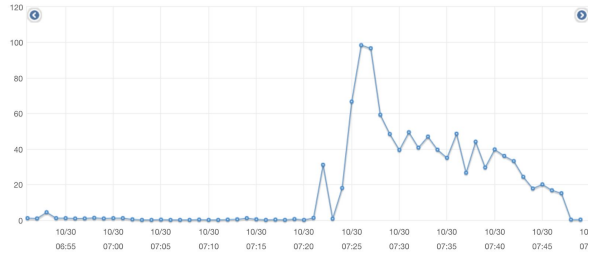
Case 3 : 128 threads against a single server

StepCounterLoadGenerator

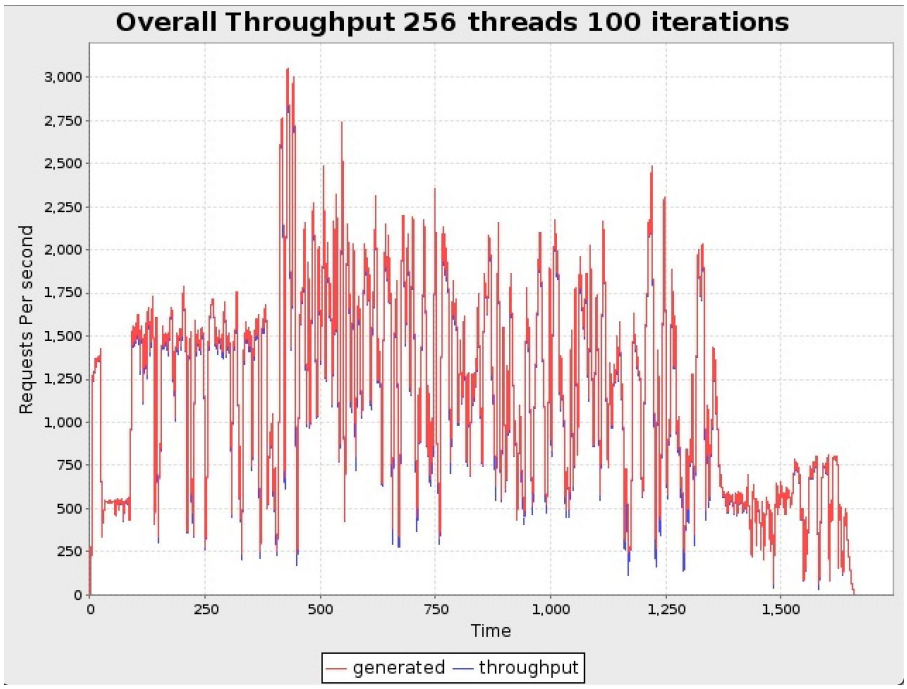
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...
Client starting.... Time: 1540853680259
WARMUP: All threads(12) running....
WARMUP complete: Time 118.978 seconds
LOADING: All threads(64) running....
LOADING complete: Time 280.433 seconds
PEAK: All threads(128) running....
PEAK complete: Time 1158.296 seconds
COOLDOWN: All threads(32) running....
COOLDOWN complete: Time 236.792 seconds
==========================================
Total number of requests sent: 962000
Total number of Successful responses: 960962
Test Wall Time: 1794.503 seconds
Overall throughput across all phases: 536.081578019095 rps.
P95 Latency = 117 ms.
P99 Latency = 121 ms.

Process finished with exit code 0
```



Case 4 : 256 threads against a single server
100% cpu at peak (compared to 20-25% peak when load balanced across 6 hosts)



WARMUP: All threads(25) running....
WARMUP complete: Time 77.56 seconds
LOADING: All threads(128) running....
LOADING complete: Time 275.453 seconds
PEAK: All threads(256) running....
PEAK complete: Time 1027.309 seconds
COOLDOWN: All threads(64) running....
COOLDOWN complete: Time 274.085 seconds
==========
Total number of requests sent: 1925500
Total number of Successful responses: 1895323
Test Wall Time: 1654.409 seconds
Overall throughput across all phases: 1163.8597227166922 rps.
P95 Latency = 72 ms.
P99 Latency = 73 ms.



Step 4 Plots and statistics for the 4 tests run against your load balanced servers. Briefly state your load balancer setup and compare the results against those you produce for step 3.

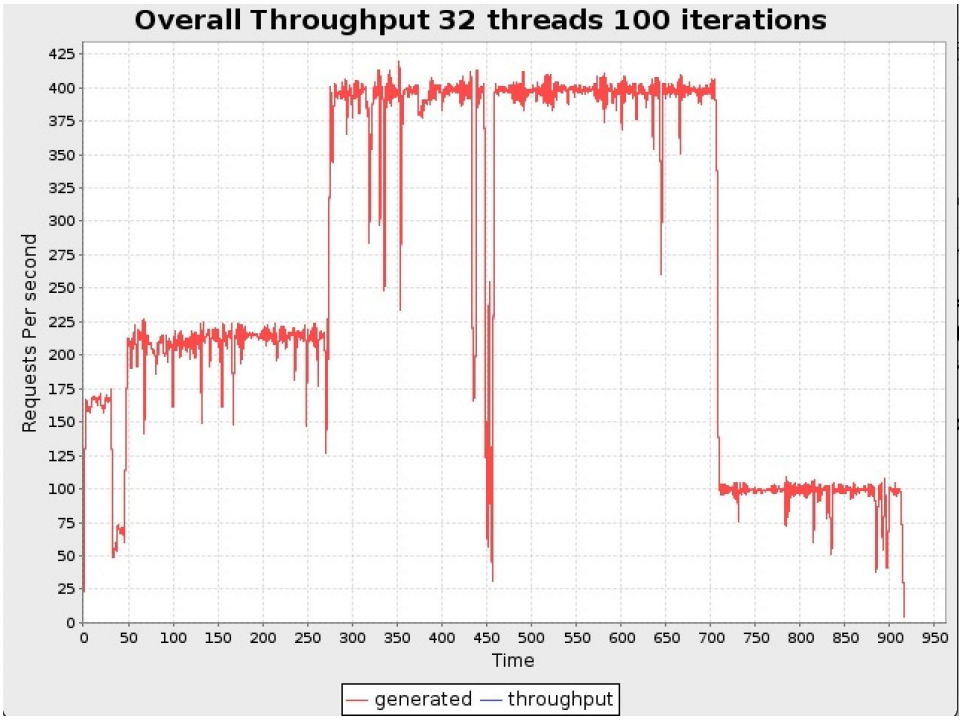
Setup for load balancer testing ->

Using Image (AMI) made from single ec2 instance and auto-scaler group to launch 4-6 instances (depends on cpu util threshold ~ 15-20%) attached to a load balancer which does health checks (at new end point /health) to verify if server is in service and forwards requests to 8080 port of other instances
each having default tomcat threads = 200
connection pool size to rds in war file = 50

Cmd ->
java -jar bsdsassignment2-stress-jar-with-dependencies.jar -t 256 -h "bsdsassignment2-load-balancer-2101220617.us-west-2.elb.amazonaws.com" -g StepCounterThroughput_256_lb.jpeg

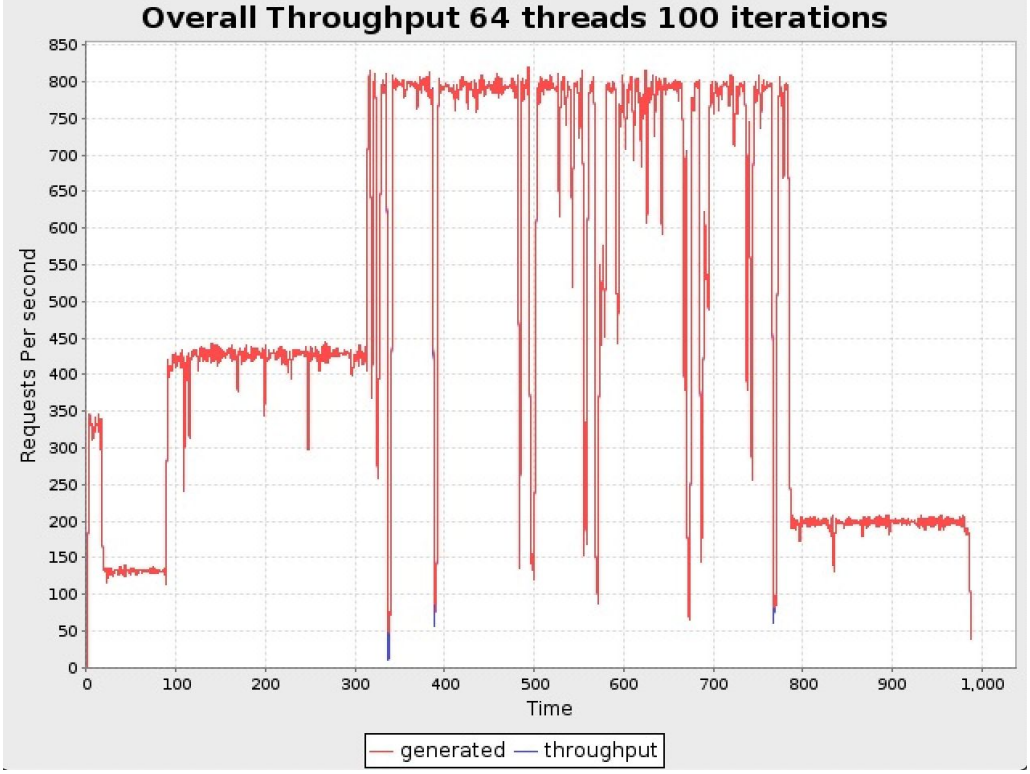
1) 32 threads and 100 Iterations on Load Balancer

Client starting.... Time: 1540878283815
WARMUP: All threads(3) running....
WARMUP complete: Time 42.13 seconds
LOADING: All threads(16) running....
LOADING complete: Time 208.769 seconds
PEAK: All threads(32) running....
PEAK complete: Time 457.928 seconds
COOLDOWN: All threads(8) running....
COOLDOWN complete: Time 205.552 seconds
==========
Total number of requests sent: 240500
Total number of Successful responses: 240498
Test Wall Time: 914.382 seconds
Overall throughput across all phases: 263.0191757930493 rps.
P95 Latency = 89 ms.
P99 Latency = 93 ms.



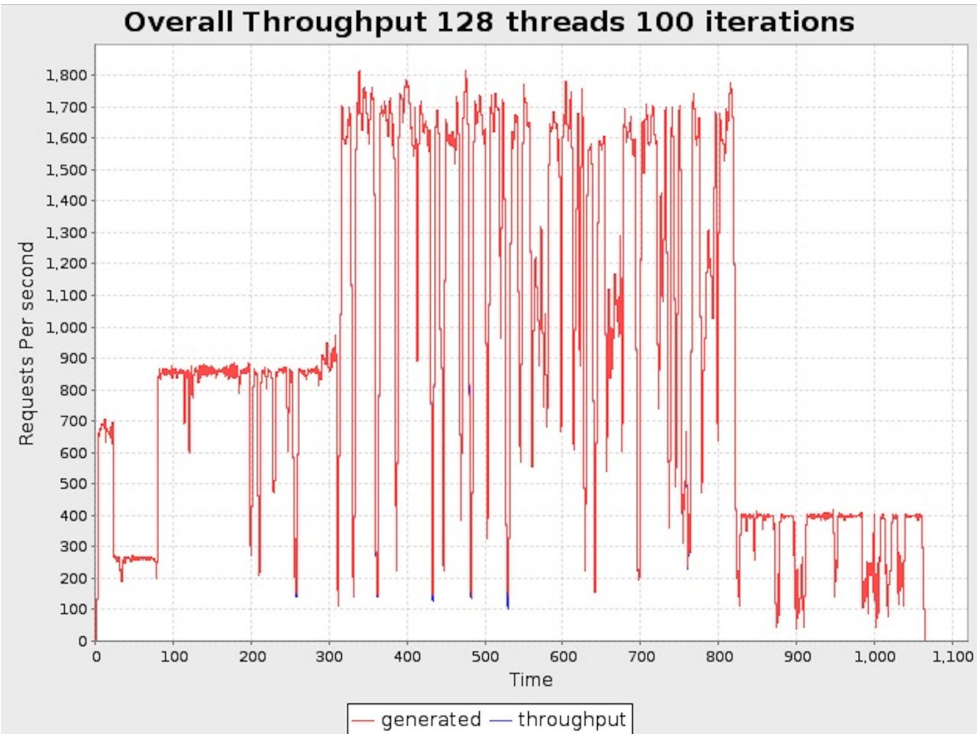
2) 64 threads and 100 Iterations on Load Balancer

Client starting.... Time: 1540877173431
WARMUP: All threads(6) running....
WARMUP complete: Time 80.75 seconds
LOADING: All threads(32) running....
LOADING complete: Time 206.761 seconds
PEAK: All threads(64) running....
PEAK complete: Time 498.199 seconds
COOLDOWN: All threads(16) running....
COOLDOWN complete: Time 201.153 seconds
==========
Total number of requests sent: 481000
Total number of Successful responses: 480857
Test Wall Time: 986.865 seconds
Overall throughput across all phases: 487.40202560633924 rps.
P95 Latency = 124 ms.
P99 Latency = 126 ms.



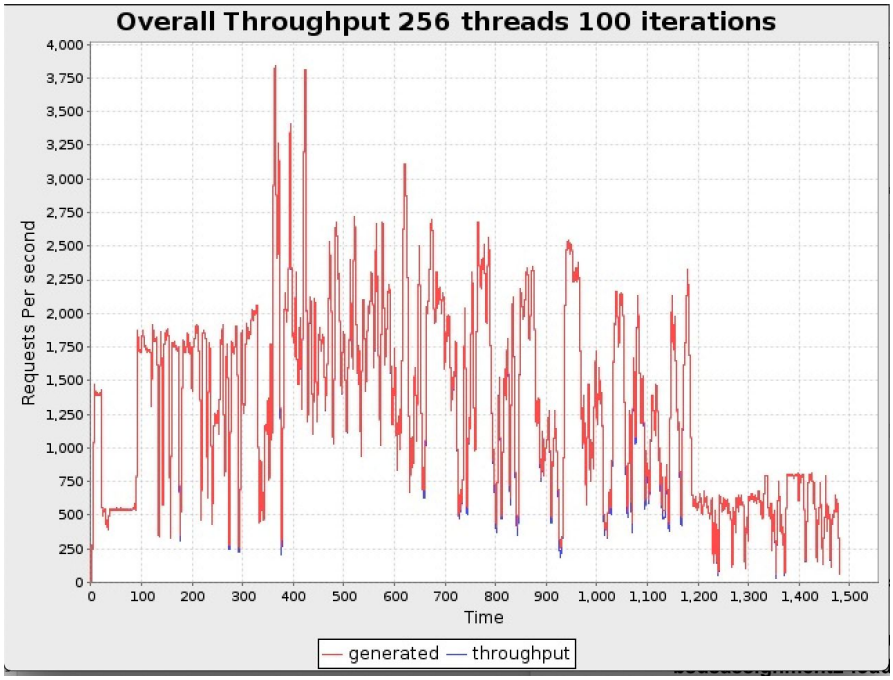
3) 128 threads and 100 Iterations on Load Balancer

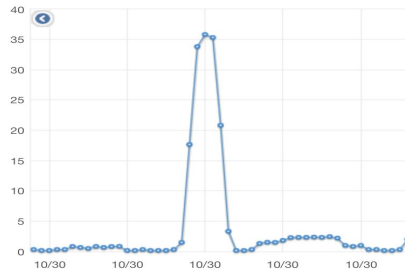
Client starting.... Time: 1540875974661
WARMUP: All threads(12) running....
WARMUP complete: Time 70.569 seconds
LOADING: All threads(64) running....
LOADING complete: Time 217.18 seconds
PEAK: All threads(128) running....
PEAK complete: Time 537.32 seconds
COOLDOWN: All threads(32) running....
COOLDOWN complete: Time 237.739 seconds
==========
Total number of requests sent: 962000
Total number of Successful responses: 961596
Test Wall Time: 1062.81 seconds
Overall throughput across all phases: 905.147674560834 rps.
P95 Latency = 119 ms.
P99 Latency = 123 ms.



4) 256 threads and 100 iterations on Load Balancer

Client starting.... Time: 1540874205758
WARMUP: All threads(25) running....
WARMUP complete: Time 80.117 seconds
LOADING: All threads(128) running....
LOADING complete: Time 232.35 seconds
PEAK: All threads(256) running....
PEAK complete: Time 877.022 seconds
COOLDOWN: All threads(64) running....
COOLDOWN complete: Time 287.798 seconds
==========
Total number of requests sent: 1925500
Total number of Successful responses: 1918508
Test Wall Time: 1477.289 seconds
Overall throughput across all phases: 1303.4010271517625 rps.
P95 Latency = 121 ms.
P99 Latency = 126 ms.



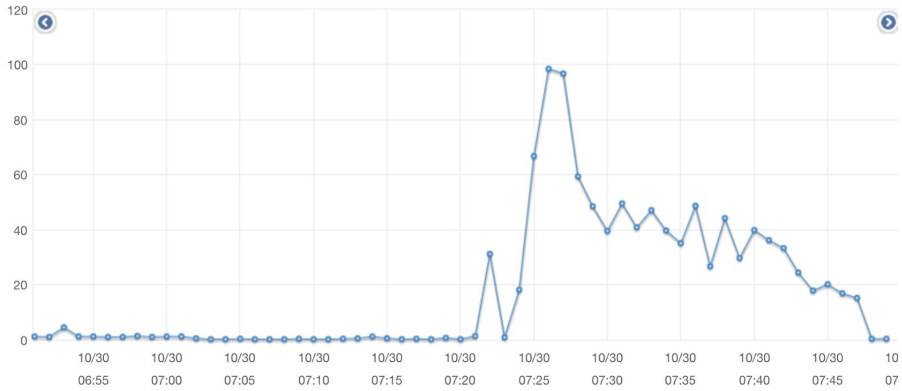


35% cpu per host across load balanced 5-6 ec2 instances

Difference in Results for Load Balancer vs Single ec2 instance testing

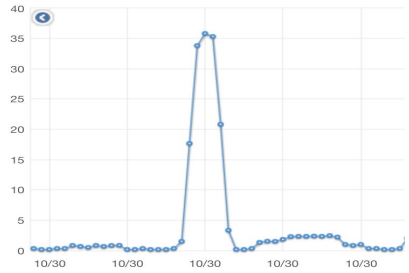
(Note :- use same client settings and same aggregated connection pooling size to RDS)

1) **Single ec2 server (t2.micro) starts bottlenecking on cpu and default heap size when run with higher number of threads (256). Heap Size was increased in tomcat server but cpu util was still reaching 100% in peak phase and there were lot of failed calls (~1.5-2%). RDS utilization at same time was very low ~20%. Even though tomcat executor was changed to use 256 threads (from default 200) but still single cpu core was not fast enough to process requests/responses from client and from downstream rds server. Hence, overall throughput achieved was also low.**




```
java.lang.OutOfMemoryError: Java heap space
Exception in thread "catalina-exec-263" java.lang.OutOfMemoryError: Java heap space
        at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
        at java.nio.ByteBuffer.allocate(ByteBuffer.java:335)
        at org.apache.catalina.connector.InputBuffer.<init>(InputBuffer.java:160)
        at org.apache.catalina.connector.InputBuffer.<init>(InputBuffer.java:147)
        at org.apache.catalina.connector.Request.<init>(Request.java:251)
        at org.apache.catalina.connector.Connector.createRequest(Connector.java:889)
        at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:307)
        at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:800)
        at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:66)
        at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:800)
        at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1471)
        at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
        at java.lang.Thread.run(Thread.java:748)
```

2) On load balancer setup, with higher number of threads, load gets distributed across different server cpu cores (or n/w card etc) (6 cores) and each cpu util was around 25-35% at peak phase. This was able to push more requests to RDS and cpu util for RDS was around 40% on same settings compared to 20% from single ec2 server. Also, failed requests percentage was low (0.3-0.4%). Overall throughput achieved was also high (~1303) compared to (~1100) from single ec2 instance. Majority difference was seen across time taken to complete loading and peak phases for 128/256 threads as those are the ones starting to push the limits of single ec2 instance. Load balancer tests took (15-20%) less time in loading and peak phases compared to single ec2 server. Overall wall time for test was also reduced by 10-15%.



Verification Steps

Server testing ->
a) Initial testing by calling random methods and validating results are correct
b) using leak detection to verify if no connections are leaked
c) no lockups/deadlocks
d) verify from catalina logs if there are any error logs

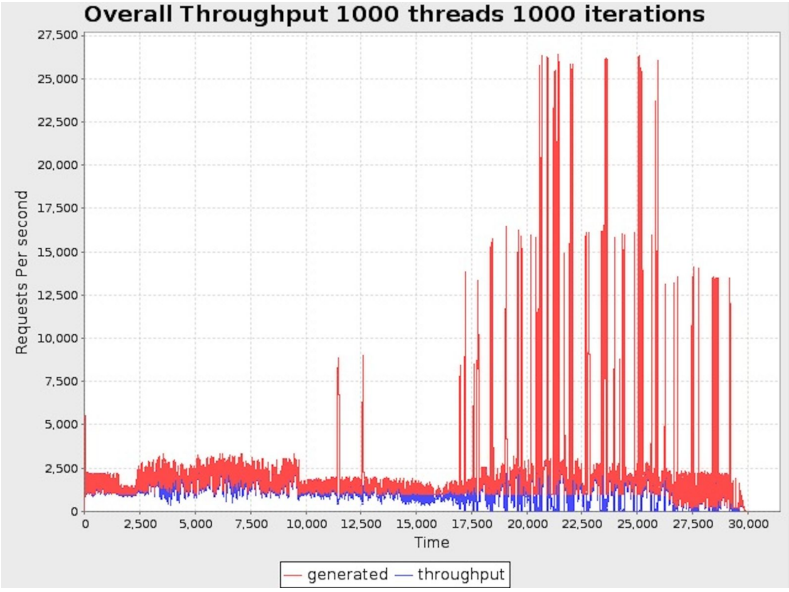
Client testing ->
a) verify each library like
StatsRecorder to verify stats are properly calculated, bucketed and aggregated
ThreadManager doing overlapping threads (no deadlocks/proper startup)
b) Verify single client thread doesn't have other overheads except sending and waiting for requests (eg. in recording stats, starting/synchronizing threads etc.)
c) failure rate is low for smaller thread settings

Client Setup ->
Timeouts - connect(1000ms), read timeout(2000ms)
Overlapping phases
Bucketed stats recorder in single thread aggregated later with other threads to avoid synchronization overhead

Step 5 : Bonus point (1000 threads for 1000 iterations)

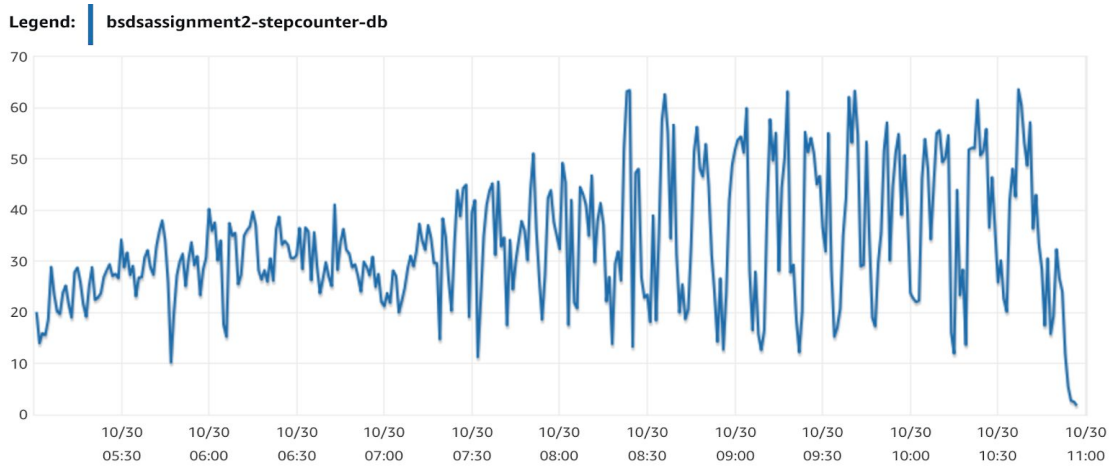
java -jar bsdsassignment2-stress-jar-with-dependencies.jar -h "bsdsassignment2-load-balancer-2101220617.us-west-2.elb.amazonaws.com" -t 1000 -n 1000 -g results/StepCounterThroughput_1000_lb.jpeg

Client starting.... Time: 1540892205991
WARMUP: All threads(100) running....
WARMUP complete: Time 1518.312 seconds
LOADING: All threads(500) running....
LOADING complete: Time 8153.439 seconds
PEAK: All threads(1000) running....
PEAK complete: Time 17151.89 seconds
COOLDOWN: All threads(250) running....
COOLDOWN complete: Time 3107.146 seconds
==========
Total number of requests sent: 75250000
Total number of Successful responses: 39782937
Test Wall Time: 29930.804 seconds
Overall throughput across all phases: 2514.1322632028196 rps.
P95 Latency = 127 ms.
P99 Latency = 129 ms.



RDS DB connections - 300

RDS CPU around 70%



Maybe bound by write iops on RDS chosen setup